TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky, informatiky a mezioborových studií

BAKALÁŘSKÁ PRÁCE

2012 Tomáš Němeček

TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky, informatiky a mezioborových studií

Studijní program: B 2612 – Elektrotechnika a informatika

Obor: 1802R007 – Informační technologie

Ovládání mobilní platformy pomocí OS Android

Mobile platform controlling using OS Android

Bakalářská práce

Autor: **Tomáš Němeček**

Vedoucí práce: Ing. Jan Strnad

Konzultant: doc. Mgr. Ing. Václav Záda, CSc.

V Liberci 17. května 2012

Prohlášení

Byl(a) jsem seznámen(a) s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval(a) samostatně s použitím uvedené literatury a na základě konzultací s vedoucím bakalářské práce a konzultantem.

Datum

Podpis

Poděkování

Na tomto místě bych rád poděkoval panu Ing. Janu Strnadovi za vedení mé práce a za poskytnutí mnoha cenných rad. Dále bych chtěl poděkovat panu doc. Mgr. Ing. Václavu Zádovi, CSc. za umožnění přístupu do laboratoře a dlouhodobého zapůjčení veškerého potřebného vybavení a panu Ing. Přemyslu Svobodovi za pomoc při natáčení prezentačního videa.

Abstrakt

Tématem bakalářské práce je ovládání mobilní platformy iRobot Create pomocí mobilního zařízení s nainstalovaným operačním systémem Android. Prvním řešeným bodem je návrh komunikačního protokolu. Protokoly byly v práci navrženy dva - XML protokol a číselný protokol. Kvůli omezení ve výkonu mobilního telefonu a složitosti implementace byl nakonec nasazen číselný protokol. Dále bylo nutné navrhnout aplikaci pro samotné ovládání robota s využitím hardwarových prostředků mobilního telefonu. Výsledná aplikace umožňuje robota ovládat jak tlačítky na displeji, tak naklápěním telefonu. Do aplikace byla navíc přidána obrazová data z kamery umístěné na robotu. Robota je tak možné ovládat a zároveň sledovat jeho polohu na displeji telefonu.

Abstract

The theme of the thesis is controling of iRobot Create mobile platform using mobile device with Android installed. The first point was to solve communication protocol. Two protocols were designed - the XML protocol and the number protocol. Due to limitations in the performance of mobile phone and complexity of implementation the protocol number was deployed. Then it was necessary to design an application to control the robot itself using the hardware resources of mobile phone. The resulting application allows user to control the mobile platform by using buttons on the screen and tilt of the phone. To the application was also added image stream from camera located on the robot. It is now possible to control and monitor position of the robot in real time.

Klíčová slova

Android, mobilní zařízení, iRobot, mobilní platforma, komunikační protokol, hardwarové prostředky

Keywords

Android, mobile device, iRobot, mobile platform, communication protocol, hardware devices

Obsah

Seznam obrázků

Seznam tabulek

Seznam zdrojových kódů

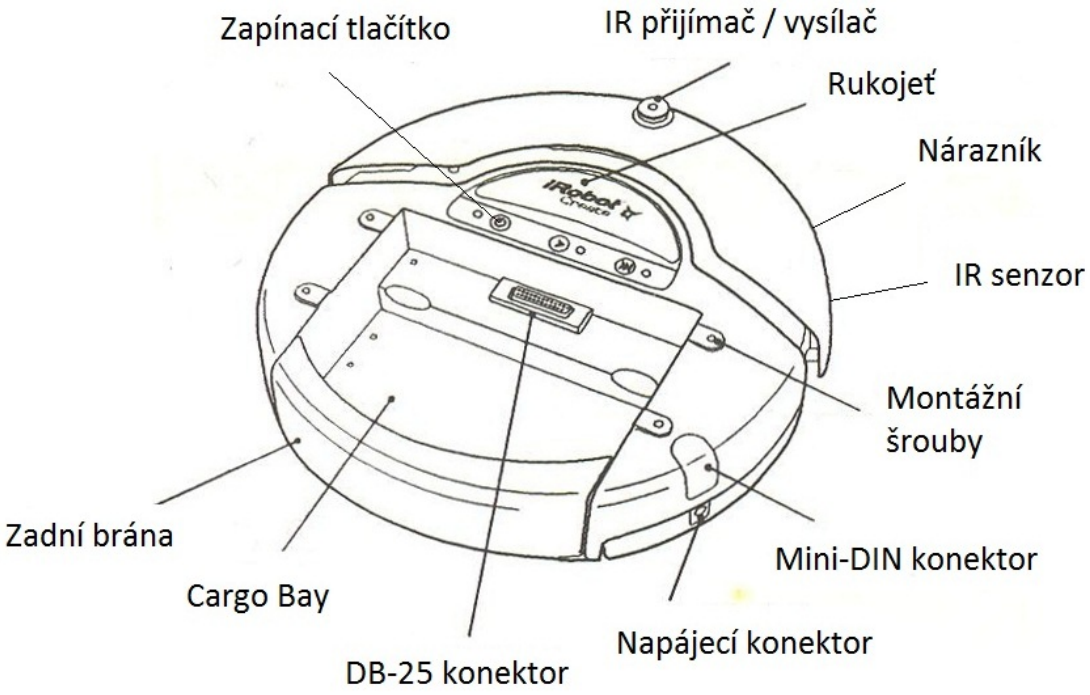
1 Úvod

Tato bakalářská práce se zabývá způsobem ovládání robotické platformy, konkrétně platformy iRobot Create, pomocí mobilního operačního systému Android. Platforma iRobot Create byla zvolena zejména z toho důvodu, že tato bakalářská práce navazuje na práci započatou v rámci bakalářského projektu. Ten se zabýval vytvořením knihovny pro ovládání právě této platformy. Byl tak nahrazen původní způsob ovládání, který spočíval v zasílání sekvencí bajtových příkazů způsobem, který umožňuje s robotem pracovat jako s objektem v jazyce C#. Kromě knihovny byl vytvořen i program pro ovládání robota z počítače přes bezdrátovou síť WiFi. Jelikož se však pro systém Android programuje v jiném programovacím jazyce, bylo potřeba některé části kódu upravit a doplnit o nové funkce. Projekt však poskytl dobrý základ.

Motivací pro bakalářskou práci bylo nejen pokračovat ve výše zmíněném projektu, ale i demonstrovat využití mobilního telefonu s nainstalovaným systémem Android pro ovládání mobilního robota. Hardware telefonu a systém Android nabízí mnoho prostředků, které lze pro ovládání využít. Například displeje dnešních mobilních telefonů, které dosahují při malé úhlopříčce velkého rozlišení, lze využít pro zobrazování dat, zobrazování obrazu, nastavování údajů pomocí dotyků na displeji či řízení robotů. Většina současných zařízení je vybavena mnoha senzory jako jsou gyroskop, akcelerometr nebo GPS modul pro sledování polohy. Využití těchto senzorů v robotice je velice různorodé. Je například možné za pomocí gyroskopu a akcelerometru ovládat rychlost a směr jízdy robota nebo robotického ramena, využít senzor osvětlení pro zapnutí pomocného osvětlení, mikrofon pro ovládání hlasem nebo například kameru pro snímání okolního prostředí.

2 Platforma iRobot Create

Platforma iRobot Create je učena zejména pro výukové účely. Vychází z platformy robotického domácího vysavače iRobot Roomba, od kterého se liší především absencí modulu pro vysávání a přítomností konektorů umožňující komunikaci se zařízením.



Obrázek 1: Popis platformy iRobot Create

2.1 Hardwarové možnosti platformy iRobot Create

Pro komunikaci s okolím je robot vybaven dvěma konektory. Konektor Mini-DIN je umístěn na pravé straně robota a jedná se o 7 pinový konektor, jehož tři piny jsou určeny pro sériovou komunikaci a zbylé čtyři slouží jako vývod napájení ± 5 V. Konektor DB-25 je umístěn v zadní části na vrchní straně robota a jedná se o 25 pinový konektor zprostředkovávající konektivitu k externím periferiím. Jsou zde jak analogové, tak i digitální vstupně-výstupní piny a vývody napájení.

Robot je současně vybaven i několika senzory. Na přední straně je nárazník, který je vybaven dvěma dotykovými čidly na každé straně. Je tak možné určit, zda robot narazil

na překážku na pravém či levém boku. Na pravé straně nárazníku je pod drobným otvorem infračervený senzor vzdálenosti s poměrně krátkým dosahem 10 cm. Ten je využíván především v aplikacích, kdy robot sleduje stěnu či objíždí překážku. Aby se nestalo, že robot sjede ze schodů či spadne do hlubokého výmolu, jsou na spodní straně umístěné dva tzv. „cliff sensors", což se dá česky přeložit jako senzory útesu, které jsou umístěny na přední straně. Robot tak při jízdě vpřed do prohlubně nespadne, při jízdě dozadu však nebezpečí hrozí. Zejména z tohoto důvodu není výrobcem doporučeno jezdit robotem směrem vzad.

V rámci projektu byl robot obohacen o perforovanou kovovou konstrukci a řídící netbook. Právě skrze něj je nyní ovládán. Je tedy možné robota ovládat i na dálku díky přítomnosti konektivity k sítím WiFi nebo Bluetooth. Pokud bude v práci zmíněno zasílání příkazů pro robota, v praxi bude příkaz zaslán na netbook, který komunikaci s robotem zprostředkuje.

3 Operační systém Android

Systém Android je dnes bezesporu nejprogresivněji se šířícím operačním systémem pro mobilní zařízení na světě. Za jeho vývojem stojí společnost Google, která v roce 2005 odkoupila tehdy takřka neznámou společnost Android Inc. a jí vyvíjený operační systém. Systém je navržen jako open source, a je proto nasazován jak malými a neznámými výrobci, tak i velkými firmami jako např. Samsung, LG, HTC, Asus, Acer, Sony atd. Každý výrobce si jej může upravit dle svého, přidat aplikace či jen změnit základní vzhled systému.

Obrázek 2: Logo operačního systému Android

3.1 Architektura systému Android

Architektura systému Android je založena na linuxovém jádře. Jádro zajišťuje ovladače pro hardware, správu paměti systému, správu napájení a součinnost procesů a služeb. Nad jádrem jsou pak umístěny knihovny jádra a knihovny jazyka Java, který je základním programovacím jazykem Androidu. Knihovny vychází v první řadě z jazyka Java SE doplněné o některé standardní knihovny, jako například Apache, a knihovny speciálně upravené pro Android. Mezi těmi jsou knihovny pro práci s SQL databázemi či OpenGL grafikou.

O spuštění programu se stará virtuální stroj Dalvik, který je speciálně navržený pro aplikace v mobilních zařízeních. Oproti virtuálnímu stroji společnosti Oracle, používaném ve standardní verzi jazyka Java, produkuje násobně menší a optimalizovanější programy. Dalvik k běhu aplikací využívá aplikační framework, který zajišťuje správu spuštěných aplikací, získávání údajů ze senzorů, komunikací s periferiemi či například správu zobrazení. Pro

popis grafických prvků, jejich rozložení a specifikaci nastavení systému a projektů, je využit jazyk XML. Aplikace lze psát i v jazyce C či C++ [2], a to díky přítomnosti standardních knihoven pro práci s těmito jazyky.

3.2 Vývoj programu pro OS Android

Aby bylo možné programovat aplikace pro systém Android, je nutné mít na počítači nainstalovaný Android SDK (Software Development Kit), což je balík nástrojů umožňujících programování. Ty se liší podle toho, pro jakou verzi systému bude aplikace určena. Po instalaci SDK je možné pracovat s projekty pomocí příkazové řádky. Android však k příkazové řádce nabízí i alternativu v podobě ADT (Android Development Tools) doplňku pro vývojové prostředí Eclipse. V následujících kapitolách budou krátce popsány všechny hlavní součásti vývojových prostředků. Balík SDK i plugin ADT je dostupný na stránkách:

http://developer.android.com/sdk/index.html

3.2.1 Android SDK

Balíček SDK je základním stavebním kamenem pro programování aplikací pro systém Android. Obsahuje základní aplikace pro vytváření projektů, emulátorů či aplikace napomáhající programátorům jako je grafický editor komponent.

3.2.2 Využití příkazové řádky

Android není standardně dodáván s vývojovým prostředím. Je však možné využít příkazovou řádku a projekty vytvářet, měnit, ladit a spouštět právě její pomocí. Základní syntaxe příkazů je následující:

android { globální parametry } action { parametry akce }

Pomocí tohoto příkazu lze v první řadě vytvářet nové virtuální stroje či měnit jejich

parametry. Druhou zásadní funkcí je práce s projekty. Je možné projekty vytvářet, upravovat, ladit, kompilovat, přidávat do nich soubory a knihovny a mazat je.

3.2.3 ADT Plugin pro vývojové prostředí Eclipse

ADT plugin je doplněk pro vývojové prostředí Eclipse, který umožňuje všechny operace, realizované příkazovou řádkou, provést pohodlněji pomocí grafického uživatelského rozhraní. ADT plugin obsahuje několik aplikací, které lze při přípravě projektu využít:

- Traceview – umožňuje zobrazit a sledovat procesy a aplikace spuštěné v telefonu nebo emulátoru včetně využití systémových prostředků
- android – aplikace zprostředkovávající možnosti SDK
- Hierarchy Viewer – umožňuje vizualizovat strukturu grafického návrhu aplikace
- Pixel Perfect – umožňuje blíže prozkoumávat grafické rozhraní aplikace na základě analýzy snímku obrazovky
- DDMS – program umožňující ladění a krokování aplikací, záznam obrazovky, analýzy vláken a systémových souborů

Kromě těchto aplikací jsou do programu Eclipse integrovány i některé funkce vylepšující editaci souborů XML:

- Graphical Layout Editor - umožňuje editaci a návrh grafického uživatelského rozhraní aplikace. Kromě editace v textovém editoru je dostupné i grafické prostředí, které umožňuje vybírat grafické komponenty z palety komponent a přidávat je na plochu aplikace
- Adnroid Manifest Editor – umožňuje v textovém i grafickém módu editovat soubory manifestu aplikace
- Menu Editor – grafický a textový editor položek menu v aplikaci

3.2.4 Emulátor systému

Mezi největší výhody ADT pluginu, resp. Android SDK, bezesporu patří emulátor systému. Tento emulátor, na rozdíl od emulátoru pro konkurenční operační systém Windows Phone, neemuluje pouze vytvořenou aplikaci, ale je schopen emulovat kompletní chod systému Android. Je tak možné okamžitě vidět výsledek práce i když není k dispozici fyzicky připojené zařízení. Prostředí emulátoru je ukázáno v příloze C.

Nastavení emulátoru disponuje širokou škálou možností. V první řadě je možné nastavit, jaká verze systému Android bude emulována. To přináší velkou výhodu pro programátory v tom, že si mohou svojí aplikaci otestovat na všech dostupných verzích systému Android a předejít tak pozdějším problémům s nekompatibilitou aplikací. Kromě verze systému lze specifikovat i hardware virtuálního zařízení, ve kterém je emulovaný systém spuštěn. Tímto způsobem se dá například určit velikost paměťové karty, rozlišení displeje, přítomnost GPS čipu a dalších senzorů či například zapnout hardwarovou akceleraci grafiky systému. V rámci jednoho počítače může být zavedeno více virtuálních zařízení a na každém je možné mít nainstalovanou jinou verzi systému. Rovněž lze mít spuštěno více emulátoru současně.

Emulátor lze spustit buď samostatně díky aplikaci AVD Manager.exe, nebo je možné využít ADT pluginu a spustit emulátor z vývojového prostředí Eclipse. ADT plugin pak může díky rozšíření DDMS s emulátorem daleko lépe pracovat. Je možné simulovat příchozí a odchozí hovory, kdy se využije mikrofon počítače a zvuk se skutečně zaznamenává, nebo odesílat a přijímat obrazové a textové zprávy. Další velmi zajímavou funkcí jsou fiktivní polohy zařízení. Je možné nastavit hodnoty senzorů (například simulovat náklon telefonu) nebo manuálně určit GPS souřadnice, na kterých se má emulátor nacházet. V neposlední řadě je zde také možnost pracovat s fotoaparátem. I v tomto případě je využit hardware počítače a jako obraz se v hledáčku telefonu zobrazuje obraz z instalované webkamery. Pokud počítač webkamerou vybaven není, zobrazují se v hledáčku emulátoru náhodně generované obrazce.

3.2.5 Komponenty aplikace

Každý projekt psaný pro OS Android sestává z několika komponent. Základní komponentou jsou aktivity. Ty jsou hlavním stavebním kamenem celé aplikace a je v nich soustředěna vět-

šina zdrojového kódu. Aktivity nemusí mít uživatelské rozhraní, ale většinou jsou využívány právě pro interakci s uživatelem a pro dodávání obsahu jsou využívány spíše služby. Jako služba je označena aplikace bez možnosti interakce s uživatelem, která je určena k dlouhodobému sledování či využívání nějakého systémového prostředku. Typickým příkladem služby je přehrávač hudby, který hraje i po tom, co uživatel aplikaci zavře, či detektor nové emailové zprávy.

4 Návrh řešení

V této kapitola bakalářské práce bude popsán návrh řešení komunikačního protokolu, grafického vzhledu aplikace pro ovládání robota a návrh využití hardwarových možností mobilního telefonu při ovládání robota.

4.1 Komunikační protokol

Hlavním úkolem komunikačního protokolu je přenos informace ze zdrojového zařízení na cílové. Ideální komunikační protokol by měl být schopen zpracovat všechny požadované funkce a být flexibilní v tom smyslu, aby bylo možné nové funkce přidávat či staré odebírat a měnit. Vzhledem k plánovanému nasazení při komunikaci přes bezdrátovou síť a velkým datovým tokem, by protokol neměl být tím, co bude celou komunikaci brzdit. Je třeba si uvědomit, že hardware mobilního telefonu není tak rychlý, aby dokázal několikrát za sekundu zpracovat složitý komunikační protokol.

4.1.1 Přenášené informace

Cílem vytvářené aplikace je přenášet příkazy a data k ovládání robota. Protokolem je nutné obsáhnout všechny příkazy, které lze robotovi zaslat a současně i od robota získat, a to i ty, které ve výsledné aplikaci potřeba nebudou. Nebude tak později problém tyto příkazy přidat. Zprávy, které lze zaslat robotovi jsou tyto:

- Příkazy ke změně rychlosti a směru jízdy
  - Globální nastavení rychlosti
  - Nastavení rychlosti pro každé kolo zvlášť
  - Nastavení úhlu, ve kterém má robot jet
- Žádost o zaslání stavu senzorů
  - Žádost o data z jednoho senzoru
  - Žádost o data ze skupiny senzorů

  - Žádost o data ze všech senzorů

- Příkaz k přehrání uložené skladby
- Příkaz k ovládání vstupně výstupních pinů konektoru DB-25
- Příkaz k zastavení či spuštění streamu obrazových dat

Ze strany robota mohou být odeslány tyto zprávy:

- Zaslání potvrzení o vykonání příkazu
- Zaslání dat ze senzoru / senzorů
- Zaslání informace o velikosti obrazových dat
- Zaslání obrazových dat
- Zaslání informace o chybě

4.1.2 Koncepce komunikačního protokolu

Možnosti, jak koncipovat komunikační protokol jsou v zásadě tři. První možností je posílat řetězce, což jsou příkazy reprezentované jinak než čísly. Druhou možností je reprezentovat příkazy právě čísly. Otázka implementace těchto dvou variant je různá. Nečíselný protokol lze zpracovat jako XML protokol, HTML protokol nebo jednoduchý protokol posílající řetězce znaků. U protokolu využívajícího čísla je pak nutné řešit pouze to, jakého datového typu čísla budou. Poslední možností koncepce protokolu je binární forma XML.

Vhledem k faktu, že funkce zasílající data přes sockety v jazycích Java a C# pracují s osmibitovými čísly, je nutné u komunikačního protokolu myslet i na důležitou roli serializacice a deserializacice dat. To se týká hlavně protokolu, který implicitně nepracuje s čísly. Uvážíme-li možnost, že bude protokol zpracováván mobilním telefonem, je toto hledisko velmi důležité.

4.1.3 Protokol nepracující s čísly

V rámci pracoviště, kde bakalářská práce vznikala, bylo rozhodnuto o návrhu komunikačního protokolu postaveném na XML. Jelikož každý robot má své vlastní příkazy a svůj vlastní způsob ovládání, musel protokol splňovat zásadní podmínku – obecnost. Tuto vlastnost je možné považovat za výhodu v tom smyslu, že protokol pak může sloužit více zařízením. Nicméně nevýhodou je složitá implementace. Prvním návrhem byla velmi jednoduchá struktura XML dokumentu zobrazená na zdrojovém kódu 1.

Zdrojový kód 1: Základní struktura XML protokolu

```
1  <head>
2    <id_command>ID</id_command>
3  </head>
4  <body>
5    <command>COMMAND</command>
6    <parameter1>PARAMETER</parameter1>
7    <parameter2>PARAMETER</parameter2>
8    .
9    .
10   <parametern>PARAMETER<parametern>
11 </body>
```

Tato struktura zůstala v podstatě zachována i ve finálním návrhu protokolu, který je graficky znázorněn v příloze A. Bylo však nutné doplnit některé důležité parametry. Do hlavičky tak přibyly, kromě id, informace o verzi protokolu, priorita či informace o opakování. Tělo bylo rozepsáno dle toho, jaké parametry se mohou v protokolu objevit. Vznikly tak samostatné položky s informacemi o stavu robota, rychlosti a směru, datech ze senzorů a obrazových datech. Protokol se tak stal mnohem univerzálnější, ale zároveň také složitější.

4.1.4 Protokol pracující s čísly

Jako alternativu k XML protokolu bylo nutné navrhnout jednodušší a lehce implementovatelný protokol, který by mohl být využit, pokud by byla rychlost komunikace pomocí XML protokolu nízká, nebo by jedno ze zařízení XML nepodporovalo. Pro zjednodušení celého

komunikačního provozu byl proto navrhnut protokol s konstantní délkou zpráv - konkrétně 5 bajtů. První bajt vždy označuje, o jaký příkaz se jedná. Zbylé bajty pak nesou samotná data. V tabulce 1 je protokol podrobně rozepsán. Pokud je záznam široký dva sloupce, značí to, že originální hodnota je datového typu short (velikost 2B). Pokud je šířka čtyři sloupce, je datovým typem Integer (velikost 4B). Prázdná pole pak označují nevyužité bajty.

4.2 Návrh aplikace pro ovládání robota

Aplikace pro ovládání má několik základních funkcí. První funkcí je komunikace s aplikací, která robota ovládá, tedy navázání, udržování a ukončení spojení. Dalšími funkcemi je nastavení rychlosti a směru jízdy robota. Jelikož je počítač na straně robota vybaven webovou kamerou, měla by zároveň tato aplikace být schopna iniciovat spojení s webovou kamerou a zobrazit obraz na displeji.

4.2.1 Grafický návrh aplikace

Přibližný a velmi zjednodušený návrh aplikace je vidět na obrázku 3.

Obrázek 3: Zjednodušený návrh aplikace

Pro ovládání směru jízdy robota budou sloužit tlačítka, která fungují jako spínače. Po stisknutí směru vpřed tedy robot jede až do momentu, kdy je stisknuto tlačítko „Stop" nebo změněn směr. Směr vzad chybí záměrně - robot není vybaven senzory pro detekci srázu pod jeho zadní hranou (viz. kapitola 2.1). Nastavení rychlosti lze ošetřit pomocí posuvníku. Rychlost robota se nastavuje v rozsahu od -500 do 500 mm/s. Je však potřeba zvolit vhodnou

Tabulka 1: Komunikační protokol

1. bajt	**Popis příkazu**	**2. bajt**	**3. bajt**	**4. bajt**	**5. bajt**
Zprávy pro robota					
1	Globální nastavení rychlosti	Rychlost			
2	Nastavení rychlosti každého kola zvlášť	Rychlost pravého kola		Rychlost levého kola	
3	Nastavení úhlu jízdy	Velikost úhlu			
4	Příkaz k zaslání údajů senzoru / senzorů	0 - jeden senzor	ID senzoru		
		1 - skupina senzorů	ID skupiny		
		2 - všechny senzory			
5	Přehrání skladby	ID skladby			
6	Nastavení pinu konektoru DB-25	Číslo pinu	Hodnota		
7	Zastavení či spuštění kamery	Status			
Zprávy od robota					
8	Hodnota senzoru	0 - jeden senzor	ID senzoru	Data senzoru	
		1 - skupina senzorů	ID skupiny	Velikost dat v bajtech	
		2 - všechny senzory	Velikost dat v bajtech		
9	Velikost obrazu	Velikost v bajtech			
255	Potvrzení vykonání či nevykonání příkazu	Příkaz	„OK”, „ERR”		

velikost kroku pohybu. Krok nastavený na 1 mm je příliš jemný, protože okem pozorovatelný rozdíl v rychlosti jízdy je patrný až při změně v řádu desítek mm/s. Ale vzhledem k plánovanému využití, kdy není potřeba jemná plynulost pohybu, lze použít i větších kroků. V prostoru pro obrazová data bude zobrazen obraz z webové kamery počítače. Velikost a kvalita obrazu budou záviset na použité kameře a jejím nastavení.

Aby bylo možné plynule regulovat směr otáčení robota, musí být přidán další posuvník. Ten bude, stejně jako posuvník pro rychlost, s určitým krokem měnit směr jízdy. V krajních polohách otáčení, tj. v krajní poloze posuvníku pro směr či při stisknutí tlačítka měnícího směr, se robot otáčí kolem své osy. Jedno kolo má tak nastavený zpětný a druhé dopředný chod - obě s toutéž hodnotou. Inovovaný návrh je vidět na obrázku 4:

Obrázek 4: Inovovaný návrh aplikace

4.2.2 Obrazová data

Před samotným zobrazováním dat je nutné rozhodnout, v jakém formátu obraz bude. Musí se vzít v úvahu rozlišení, barevná hloubka a z toho vycházející datový tok. Ten je podstatný z toho důvodu, že obraz je posílán přes bezdrátovou síť WiFi. Je tedy nutné počítat s limitem maximální přenosové rychlosti za ideálních podmínek, který má u WiFi standardu b a g hodnotu 7 MB/s a u WiFi standardu n pak 75 MB/s.

Obraz je vytvářen v aplikaci psané v jazyce C#. Ten umožňuje různé formáty obrazu [8]. Mezi nejpoužívanější formáty patří BMP (Bitmap) a JPEG (Joint Photographic Experts Group). Je samozřejmě možné využít i další formáty jako GIF, Icon, PNG nebo TIFF. Každý z obrazových formátů může pracovat v několika barevných prostorech [8]. Dnes již

Tabulka 2: Výsledné datové toky

Rozlišení	**Počet bitů na pixel**	**Počet snímků za sekundu**	**Datový tok**
160 x 120 px	16 bpp	25 fps	0,9216 MB/s
160 x 120 px	24 bpp	25 fps	1,44 MB/s
320 x 240 px	16 bpp	25 fps	3,84 MB/s
320 x 240 px	24 bpp	25 fps	5,76 MB/s
640 x 480 px	16 bpp	12 fps	7,3728 MB/s
640 x 480 px	24 bpp	12 fps	11,0592 MB/s
640 x 480 px	16 bpp	25 fps	15,36 MB/s
640 x 480 px	24 bpp	25 fps	23,04 MB/s
1280 x 720 px	16 bpp	12 fps	22,1184 MB/s
1280 x 720 px	24 bpp	12 fps	33,1776 MB/s
1280 x 720 px	16 bpp	25 fps	46,08 MB/s
1280 x 720 px	24 bpp	25 fps	69,12 MB/s

standardním barevným prostorem je 24 bitový RGB barevný prostor, kdy je každá z barev tvořena 8 bity (může tedy nabývat hodnot 0 až 255). Dalšími barevnými prostory jsou pak 32, 48 a 64 bitové barevné prostory u nichž je jedna barva tvořena 8 resp. 16 bity. U všech RGB barevných prostorů je možné přidat i alfa kanál, neboli průhlednost. Barevný prostor může být rovněž 16 bitový šedotónový.

Datový tok obrazových dat v nekomprimovaném formátu BMP lze vypočítat dle jednoduchého vzorce:

datový tok = (šířka obrazu) ∗ (výška obrazu) ∗ (počet bitů na pixel) ∗ (počet snímků za sekundu)

Z výsledných hodnot v tabulce 2 je patrné, že kdybychom neměli k dispozici WiFi starndartu n, ale pouze pomalejší b či g, a komunikovali maximální možnou rychlostí, byli bychom

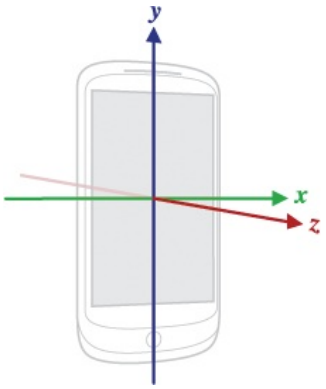
schopní přenést obraz s maximálním rozlišením 320 x 240 px. Při využití formátu JPEG bude velikost přibližně čtvrtinová. Bude tak možné přenést obrázek s rozlišením až 640 x 480 px.

4.3 Návrh využití hardwarových prostředků mobilního telefonu

Všechny dnes prodávané mobilní telefony jsou po hardwarové stránce velmi dobře vybaveny. Kromě rozměrného displeje s velkým rozlišením obsahují i řadu senzorů - gyroskop, akcelerometr, GPS modul, senzor osvětlení a mnoho dalších. Pro aplikaci ovládání robota se nabízí využití již zmíněného gyroskopu - senzoru náklonu. S jeho pomocí bude možné robota ovládat i bez dotyku prstu na klávesnici.

4.3.1 Senzor náklonu

Senzor náklonu v systému Android pracuje ve valné většině současných telefonů se třemi osami - x, y a z. Jejich poloha je znázorněna na obrázku 5.

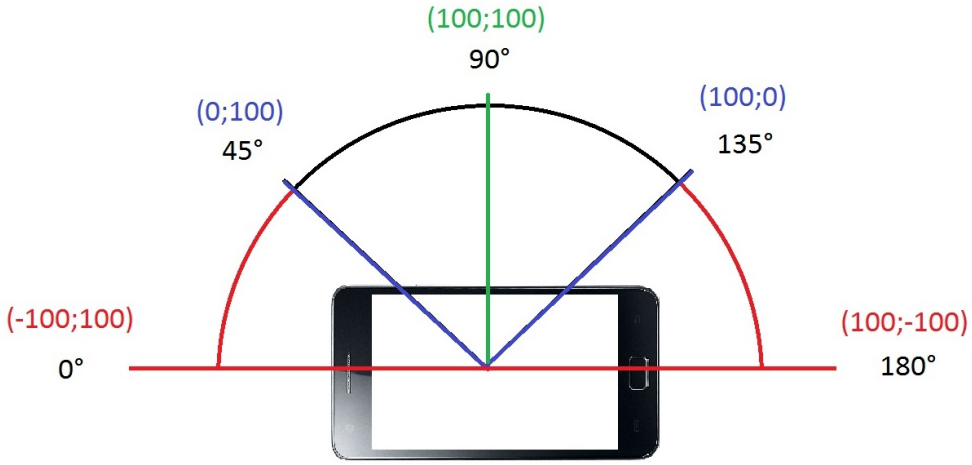


Obrázek 5: Umístění os tříosého gyroskopu

4.3.2 Ovládání směru jízdy robota

Pro ovládání směru robota bude využita rotace v ose z. Na obrázku 5 je naznačeno, jak se bude vyvíjet rychlost v závislosti na otočení telefonu. Levé číslo v závorkách značí rychlost levého kola. Číslo napravo pak představuje pravé kolo. V případě, že je telefon ve vodorovné poloze, pojedou obě kola na 100% své rychlosti. Při otočení doleva se bude rychlost levého kola zpomalovat a při dosažení úhlu 45° se levé kolo přestane otáčet úplně. Stejná situace

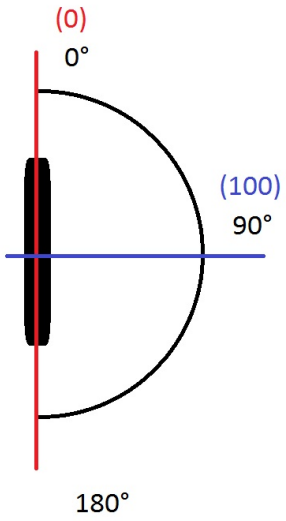
nastane i při otáčení telefonem na pravou stranu. Zde bude mezním úhlem úhel 135° a zpomalovat bude pravé kolo. Při překročení úhlu 45° na levé straně a úhlu 135° na pravé straně se robot bude otáčet kolem své osy. Kola tedy budou mít stejnou rychlost, ale jedno z nich pojede dopředu a druhé dozadu.



Obrázek 6: Ovládání směru jízdy robota

4.3.3 Ovládání rychlosti jízdy robota

V případě ovládání rychlosti bude využita rotace v ose y. Na obrázku 7 je znázorněn vztah náklonu a rychlosti. Při nulovém nebo záporném náklonu bude rychlost robota nulová a při naklánění telefonu směrem vpřed bude rychlost lineárně růst až do náklonu 90°, kdy rychlost dosáhne 100%. Tento úhel je mezním úhlem, protože po jeho překročení už se rychlost nezvyšuje.



Obrázek 7: Ovládání rychlosti jízdy robota

5 Praktické řešení

Praktické řešení aplikace pro ovládání robota se dělí na dvě části. První částí je vytvoření programu pro mobilní telefon. Tento program bude komunikovat s druhým programem, který bude spuštěn na počítači, který má robot připevněn na své konstrukci, a jež slouží k jeho ovládání. Schéma komunikace je popsáno v kapitole 5.2. První jmenovaný program bude vytvořen v jazyce Java a vývojovém prostředí Eclipse (viz. kapitola 3.2). Druhý program je napsán v jazyce C# a vývojovém prostředí Microsoft Visual Studio. Základ tohoto programu je převzat z bakalářského projektu, kde sloužil pouze jako demonstrační program. V rámci bakalářské práce byla ale většina kódu přepsána, aby vyhovovala současnému zadání.

5.1 Použitá zařízení

Při řešení bakalářské práce byl využit mobilní telefon Samsung Galaxy S2. Základní parametry tohoto přístroje jsou:

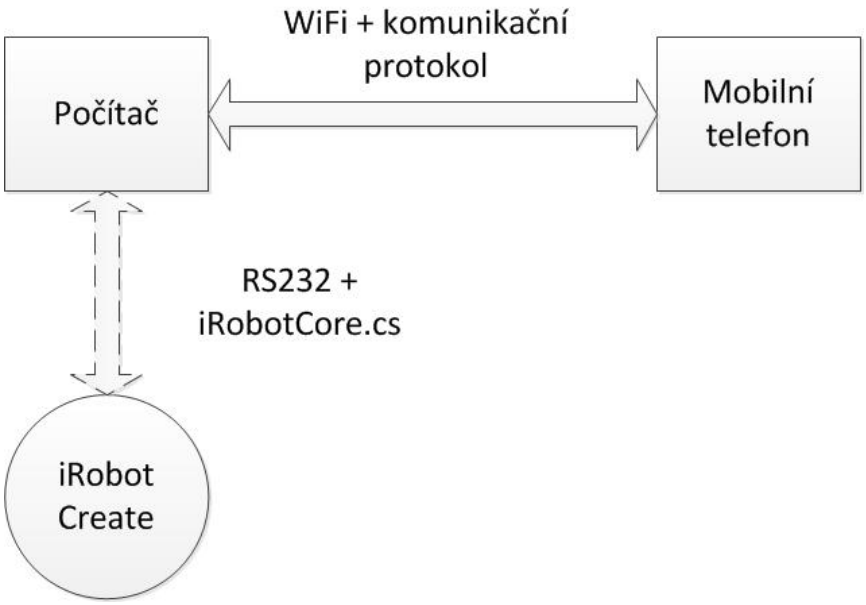
- Android ve verzi 4.0.3
- Dvoujádrový procesor s taktem 1,2 GHz
- Operační paměť o velikosti 768 MB
- Displej s rozlišením 480 x 800 px
- Vnitřní paměť 16 GB
- Tříosý gyroskop

Robot je osazen počítačem s nainstalovaným systémem Microsoft Windows 7. Jedná se o klasický netbook s procesorem Intel Atom. Tato platforma byla použita především kvůli nenáročnosti na napájení a rozměrům.

5.2 Schéma komunikace

Celý projekt je koncipován tak, že aplikace na mobilním telefonu komunikuje pomocí bezdrátové sítě WiFi s počítačem umístěným na robotu. Ten pak obstarává samotnou komunikaci

s robotem pomocí rozhraní RS 232 (s použitým převodníkem na USB). Schéma komunikace je graficky znázorněno na obrázku 8.



Obrázek 8: Schéma komunikace

5.3 Rozdělení klient - server

Komunikace přes sockety pracuje na principu klient - server. Bylo tedy nutné rozhodnout, která aplikace bude klientem a která serverem. Realizovat ovládání jednoho robota více mobilními telefony (tedy tak, že aplikace na robotu bude serverem a mobilní telefon klientem) je v praxi nevyužitelné. Naproti tomu ovládat jedním telefonem více robotů (robot je klientem a mobilní telefon serverem) už využitelné je. Rozdělení klient - server je tedy následující: aplikace na mobilním telefonu je serverem a aplikace ovládající robota je klientem.

5.4 Implementace komunikačního protokolu

Zásadní otázkou byla volba komunikačního protokolu. Na výběr byly dvě možnosti (viz. kapitola 4.1.2). V první fázi vývoje byl implementován protokol XML. Ne však jeho kompletní verze, ale jedna z raných vývojových verzí. Ta už ale obsahovala všechny možnosti sloužící k ovládání robota. Jeho implementace vyžadovala naprogramování metod pro serializaci na

jedné a deserializaci na druhé straně. Už při tomto úkolu vznikly potíže a to hlavně kvůli tomu, že protokol XML nemá konstantní velikost paketu. Data se tedy nemohla rozdělit podle šablony, ale muselo se s každým paketem zacházet zvlášť. Dalším problémem byl fakt, že jedna aplikace je programována v jazyce Java a druhá v jazyce C#. Všechny funkce tak musely být programovány dvakrát. Už při této první verzi vykazoval mobilní telefon velké zpomalení. Díky tomu, že jsou každou sekundu zpracovávána obrazová data a příkazy pro robota zároveň, bylo nad možnosti procesoru mobilního telefonu, aby XML protokol zpracoval dostatečně rychle.

Od XML protokolu bylo tedy ustoupeno a byl použit číselný protokol s konstantní velikostí paketu - 5 bajtů. Tento protokol měl výhodu ve velmi snadné implementaci a možnosti jednoduchého přidávání nových funkcí. Nevýhoda je však v tom, že protokol není dobře „čitelný". Místo slovních popisů hierarchické struktury se jedná o pole pěti za sebou jdoucích čísel. V tomto ohledu má protokol XML výhodu.

5.5 Zpracování obrazových dat

Vzhledem k datovému toku nekomprimovaného obrazu ve formátu BMP (viz. tabulka 2) byla možnost přenosu obrazu v tomto formátu již předem zamítnuta a byla zvolena komprese do formátu JPEG - konkrétně bezztrátová komprese. Na straně klienta je tak vytvořena Bitmapa, ta je následně převedena na formát JPEG, odeslána do mobilního telefonu a v něm zobrazena. Výhoda komprese dat je tedy jasná - úspora velikosti. Kompresí obrazu byla však výrazně zvýšena výpočetní náročnost klientského programu. Zatímco při posílání nekomprimovaných dat se vytížení procesoru na netbooku pohybovalo kolem 20%, při využití komprese stoupla tato hodnota na trojnásobek.

5.6 Program pro mobilní telefon

V následujících částech textu bude stručně popsána aplikace umožňující ovládání robota na dálku. Vývoj programu probíhal v několika fázích. V první fázi byly naprogramovány pouze funkce pro přenos příkazů. Program tedy uměl pouze zobrazit přijatou zprávu a jinou zprávu zase odeslat. Na této vývojové verzi byla doladěna komunikace přes sockety. Následná

verze už implementovala komunikační protokol a měla tlačítka pro ovládání robota. Díky čtyřsměrnému kříži tak bylo možné s robotem pohybovat a pomocí posuvníku nastavovat rychlost. V dalších verzích bylo implementováno přijetí obrazu. První pokusy byly však pomalé díky nepřítomnosti komprese dat a nevyřešeném ošetření chyb v přenosu. Poslední verze programu již implementovala i ovládání náklonem. Na závěr bylo ještě nutné vyřešit korektní ukončení a uspání aplikace. Pokud je totiž aplikace na Androidu přesunuta do pozadí, ve výchozím nastavení se po obnovení restartuje. Tomu bylo nutné předejít a stav aplikace je tak ukládán a následně vyvolán. Struktura aplikace je rozdělena do několika základních tříd:

- MainActivity - tato třída je hlavní třídou celého projektu, jsou v ní definovány prvky uživatelského rozhraní, ošetřeny události (např. stisk tlačítka, pohyb posuvníku), zpracováván obraz a sbírána data ze senzorů
- ServerCommunication - třída, ve které je pomocí socketu spravována bezdrátová komunikace a ve které jsou připojováni klienti
- MessageInHandler - třída zpracovávající data vstupující do aplikace
- MessageOutHandler - třída odesílající zprávy klientské aplikaci
- CameraInHandler - třída starající se o příjem obrazových dat

5.6.1 Sockety v jazyce Java

Jako prostředek k vývoji síťových aplikací se v jazyce Java používají tzv. „sockety". Jedná se o datové struktury umožňující navázat, udržovat a ukončit spojení mezi dvěma síťovými body. V této aplikaci bylo využito třídy ServerSocket [11], která zprostředkovává serverovou funkcionalitu. Má tři konstruktory:

- *public ServerSocket()*
  Vytvoří instanci třídy ServerSocket, která nemá přiřazený port ani adresu.

- *public ServerSocket(int port)*
  Vytvoří instanci třídy ServerSocket, která otevře spojení na současné síťové adrese a zadaném portu.

- *public ServerSocket(int port, int backLog)*
  Vytvoří instanci třídy ServerSocke, která otevře spojení na současné síťové adrese a zadaném portu. Proměnná backLog udává maximální počet žádostí o spojení, které má tento server obsloužit.

- *public ServerSocket(int port, int backLog, InetAddress bindAddr)*
  Vytvoří instanci třídy ServerSocket, která otevře spojení na adrese bindAddr a zadaném portu. Proměnná backLog udává maximální počet žádostí o spojení, které má tento server obsloužit.

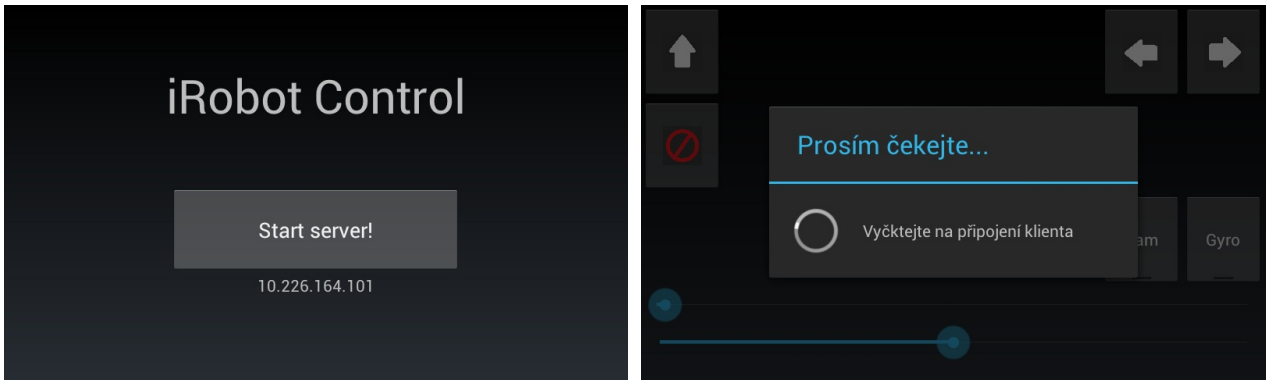
5.6.2 Vícevláknovost aplikace

Vzhledem k použití socketu bylo nutné aplikaci koncipovat jako vícevláknovou. Pokud by byla aplikace jednovláknová, nemohl by socket přijímat a odesílat zprávy ve stejný čas, ale musel by se mezi těmito dvěma úkoly neustále přepínat. Při rozdělení aplikace na vlákna je možné zároveň naslouchat příchozí komunikaci, odesílat potřebná data, zpracovávat uživatelské rozhraní nebo například skenovat žádosti o spojení se socketem.

Aplikace tedy z pohledu vláken funguje následovně: při spuštění aplikace je automaticky vytvořeno vlákno hlavní třídy. To vytvoří druhé vlákno, které skenuje žádosti o spojení se socketem. Pokud je nějaká žádost úspěšná, vytvoří se tři další vlákna - v jednom se zpracovává příchozí komunikace, v druhém odchozí komunikace a třetí vlákno zpracovává příchozí obrazová data. Všechna tato vlákna běží po celou dobu životnosti aplikace. A teprve s jejím ukončením, nebo se ztrátou spojení s klientem, zanikají. Tento proces je zachycen sekvenčním diagramem UML v příloze D.

5.6.3 Grafika uživatelského rozhraní

V aplikaci bylo nutné oddělit dva úkoly - spojení s klientem a samotné ovládání robota. Pokud by byly všechny ovládací prvky na jedné obrazovce, bylo by to matoucí a uživatel by



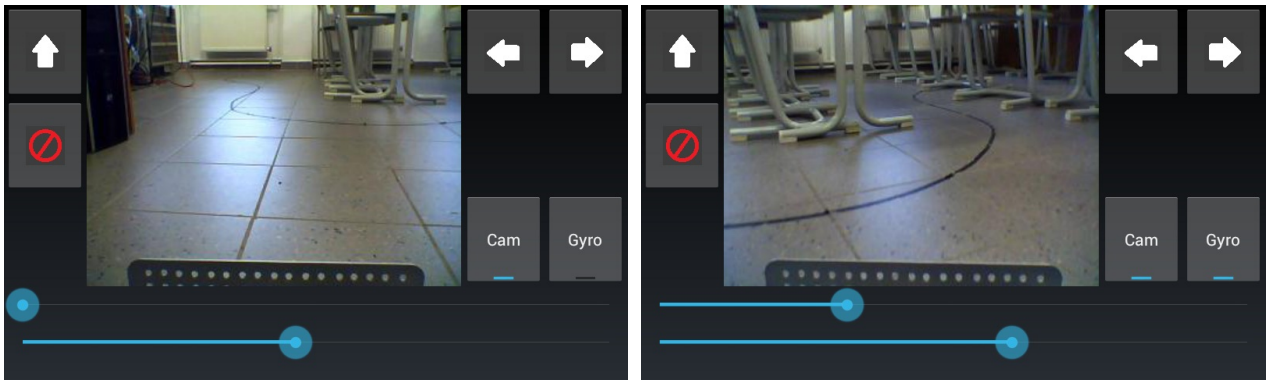
(a) Úvodní obrazovka (b) Obrazovka čekání na klienty

Obrázek 9: Vzhled úvodních obrazovek

mohl pojmout dojem, že již může robota ovládat, aniž by s ním však byl spojen.

Na první obrazovce je tak pouze uveden název aplikace, IP adresa serveru a tlačítko pro spuštění skenování nových klientů. V případě, že uživatel klikne na tlačítko pro spuštění serveru, zobrazí se dialog informující o tom, že server čeká na nové klienty. Pod tímto dialogem již jsou vidět prvky pro ovládání robota. Není však s nimi možná interakce. Konkrétní podobna úvodních obrazovek je vidět na obrázku 9.

Po úspěšném spojení s klientem je zobrazena obrazovka s ovládacími prvky. Na levé straně to jsou: tlačítko pro jízdu vpřed a tlačítko pro zastavení. Tato tlačítka fungují jako spínače. Po jejich stisknutí tedy robot jede požadovaným směrem až do doby, než je stisknuto jiné tlačítko změny směru nebo tlačítko pro zastavení. Na pravé straně v horní části jsou umístěna tlačítka pro jízdu vlevo a vpravo. Tato tlačítka způsobí otáčení robota kolem své osy. Pod nimi je spínač zobrazení obrazu z kamery a spínač pro spuštění ovládání pomocí gyroskopu. Ve spodní části jsou pak umístěny dva posuvníky. Horní slouží pro nastavení rychlosti a spodní pro změnu směru. Výchozí pozice druhého posuvníku je uprostřed, tedy přímý směr. Pokud uživatel táhne jezdcem do strany, robot se otáčí v požadovaném směru. Funkce tohoto posuvníku je totožná s ovládáním náklonem - čím více na straně je jezdec na posuvníku, tím více robot zatáčí. Při dosažení krajní polohy se robot začne otáčet kolem své osy. Příklady uživatelského rozhraní jsou vidět na obrázku 10. Na obrázku 10a stojí robot na místě a telefon je ve vodorovné poloze. Na obrázku 10b robot jede přibližně třetinovou rychlostí a mírně zatáčí doprava.



(a) Stojící robot

(b) Poloviční rychlost, směr mírně doprava

Obrázek 10: Příklady uživatelského rozhraní

5.6.4 Popis pomocí XML

Celé uživatelské rozhraní je popsáno v jazyce XML. V něm je definováno jak rozložení, tak i vzhled a chování všech prvků uživatelského rozhraní. Pro definování rozložení prvků slouží tzv. „Layouts". Těch je v Androidu hned několik:

- FrameLayout - nejjednodušší rozložení, které na obrazovce vytvoří rámec dané velikosti. V tomto rámci se všechny prvky rovnají do levého horního rohu a není možné nijak specifikovat jejich velikost. Využití má například v případech, kdy očekáváme zobrazení obrázku o dané velikosti.

- LinearLayout - v tomto rozložení se prvky rovnají vedle sebe nebo pod sebe a to v závislosti na nastavení parametru *android:orientation* , který nabývá hodnot *horizontal* či *vertical* .

- TableLayout - při použití tohoto rozložení se vytvoří tabulka s definovaným počtem řádek a sloupců, do jejíž buněk se pak prvky uživatelského rozhraní umísťují.

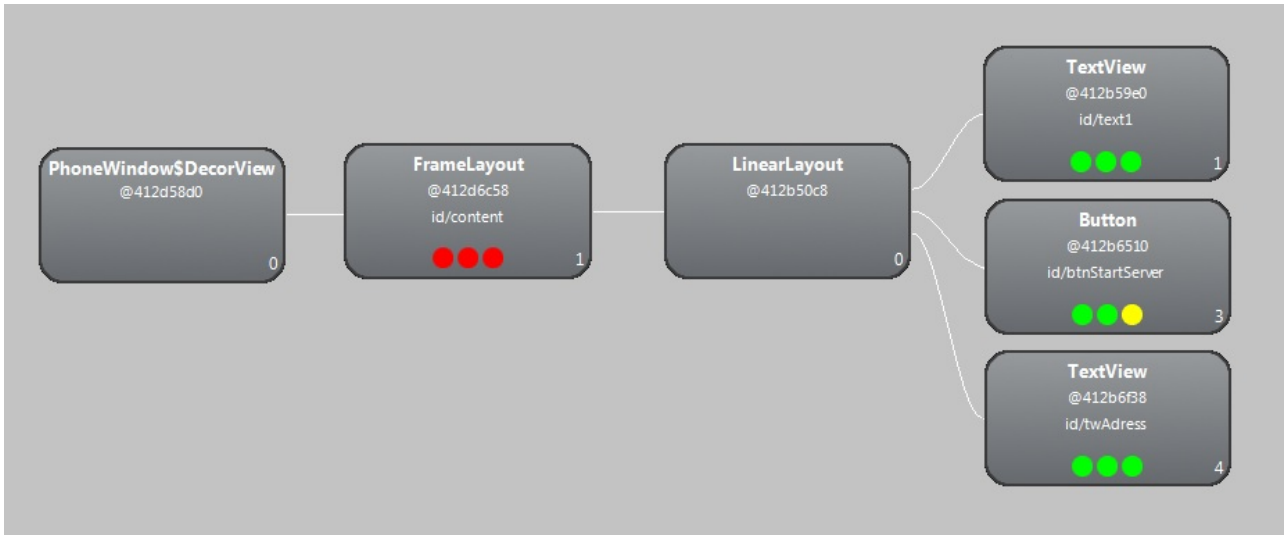
- RelativeLayout - rozložení RelativeLayout umožňuje nastavit pozici prvku v závislosti na poloze nadřazeného prvku nebo poloze jiného prvku uživatelského rozhraní. Je současně možné prvky rovnoměrně rozložit (například pokud chceme na displej umístit pouze tři tlačítka a rovnoměrně je zarovnat) pomocí parametru *android:weight* .

Jako příklad vytvoření grafického prvku v jazyce XML je uvedena definice tlačítka „STOP" ve zdrojovém kódu 2. Položka id označuje identifikátor, podle něhož je s tlačítkem v Androidu pracováno, následuje výška, šířka, obrázek na pozadí a text, který se na tlačítku zobrazí.

Zdrojový kód 2: Definice tlačítka STOP

```
1 <ImageButton
2   android:id="@+id/btnStop"
3   android:layout_width="70dp"
4   android:layout_height="80dp"
5   android:src="@drawable/stop"
6   android:text="Stop" />
```

Rozložení prvků úvodní obrazovky zachycené programem HierarchyViewer (viz. kapitola 3.2.3) je na obrázku 11. Barevná kolečka značí náročnost na vykreslování. V případě Layoutu jsou červená, protože program čeká na vykreslení dílčích prvků uživatelského rozhraní.



Obrázek 11: Rozložení prvků úvodní obrazovky

Kvůli složitosti druhého uživatelského rozhraní by bylo velmi nepřehledné přidávat stejný výpis i pro hlavní uživatelské rozhraní. To je tedy zobrazeno názorněji jako obrázek s popisy v příloze B.

5.6.5 Ovládání rychlosti a směru jízdy robota

Ovládání rychlosti je v aplikaci řízeno horním posuvníkem na uživatelském rozhraní. Ten má nastaven krok 100 dílků. Jelikož je ale robot schopen vyvinout rychlost mezi 0 až 500 mm/s, je v každém kroku posuvníku změněna rychlost o 5 mm/s. Toto rozlišení se při testování ukázalo jako ideální poměr mezi jemností a rychlostí zpracování. Pokud by byl krok posuvníku nastaven na 1 mm/s, nestíhal by program při rychlé změně zareagovat. Pokud by byl naopak krok nastaven na větší, způsobila by změna rychlostí trhaný pohyb robota.

Ovládání směru je vyřešeno nastavováním hodnot pro každé kolo zvlášť. Nenastavuje se však absolutní rychlost, ale procentuální podíl rychlosti nastavené posuvníkem. Při plné rychlosti robota tedy není odeslána hodnota 500, ale 100. Je tak řečeno, že má robot jet na 100% globálně nastavené rychlosti. Posuvník pro nastavení směru je rozdělen na 30 dílků - tedy 15 pro jízdu napravo a 15 pro jízdu nalevo. Tento počet byl opět zvolen po sérii testování. U změny směru není nutná taková jemnost jako při změně rychlosti a 15 dílků je tak postačující hodnota.

Ovládání rychlosti jízdy a směru pomocí naklánění telefonu kopíruje chování dolního posuvníku na displeji. Při natočení telefonu pro změnu směru jízdy se tak vyvolá stejná funkce jako při tažení prstem po posuvníku. I ovládání naklápěním je tedy rozděleno na 30 částí a ovládání rychlosti na 100 částí.

5.6.6 Zobrazení obrazu

Jak bylo uvedeno v kapitole 5.5, pracuje program s obrazem ve formátu JPEG. Jelikož má tento formát variabilní délku obrazu, musela k tomu být uzpůsobena funkce pro přijetí obrazu. Zároveň musely být eliminovány chyby v obraze, které vznikají z větší části na straně klientské aplikace, která obraz vytváří. Podíl na chybách však může mít i samotný přenos. Zpracování obrazu bylo v prvních verzích řešeno tak, že byla přijata informace o velikosti obrazu a následně samotná obrazová data. Toto řešení bylo funkční až do chvíle, kdy nenastala chyba přenosu nebo kdy nebyla porušena data o velikosti obrazových dat. Pak již nebylo možné rozlišit, co je velikost dat a co je obraz. Aby byly i tyto chyby odstraněny,

byla přidána startovací sekvence. Tou začíná každý přenos obrazu. Nastane-li tedy chyba přenosu, jsou zbylá data ignorována a zahazují se do doby, kdy není na vstupu opět startovací sekvence. V programu se tyto chyby projevují mírným záškubem obrazu. Jako startovací sekvence byla zvolena posloupnost čísel 127 a 20. Tato čísla byla zvolena na základě statistického přehledu a ač existuje pravděpodobnost, že se tato posloupnost v obraze objeví, při bezchybném přenosu to neovlivňuje provoz. Celý proces příjmu obrazových dat je vypsán ve zdrojovém kódu 3.

V první části funkce je přijímáno první číslo sekvence. Pokud není přijato, jsou načítána další čísla až do chvíle, než se na vstupu první číslo sekvence objeví. Následně je kontrolována přítomnost druhého čísla a v případě, že druhé číslo na vstupu není, funkce pokračuje opět od začátku. Pokud je i druhé číslo přijato, je následně načtena velikost obrazových dat. Funkce pro komunikaci mezi sockety pracují s datovým typem Byte. Je tak nutné následně poskládat pole bajtů do jedné hodnoty typu Integer. Toto je realizováno bitovou rotací. Ve chvíli, kdy je známa velikost dat, je ověřeno, zda se hodnota pohybuje v povoleném rozmezí. Pokud ano, jsou přijata samotná obrazová data.

Zdrojový kód 3: Přijetí obrazových dat

```
1  /* Příjem prvního čísla sekvence */
2  byte[] startSequence = new byte[] { 0 };
3  cameraInputStream.read(startSequence, 0, 1);
4
5  while (startSequence[0] != 127) {
6    cameraInputStream.read(startSequence, 0, 1);
7  }
8
9  /* Příjem druhého čísla sekvence */
10 byte[] startSequence2 = new byte[] { 0 };
11 cameraInputStream.read(startSequence2, 0, 1);
12
13 if (startSequence2[0] == 20) {
14   byte[] imageSize = new byte[4];
15
16   /* Přijetí pole s velikostí obrazu */
17   for (int size = 0; size < 4;) {
```

```
18        size += cameraInputStream.read(imageSize, size, imageSize.
              length - size);
19      }
20
21      /* Převedení pole hodnot na hodnotu int */
22      int value = ((imageSize[3] & 0xff) << 24) | ((imageSize[2] & 0
           xff) << 16) | ((imageSize[1] & 0xff) << 8) | (imageSize[0]
           & 0xff);
23
24      /* Kontrola zda je velikost v rámci maximálních hodnot */
25      if (!(value > 40000) && !(value < 0)) {
26        byte[] image = new byte[value];
27
28        /* přijetí obrazu */
29        for (int read = 0; read < image.length;) {
30          read += cameraInputStream.read(image, read, image.length -
                read);
31        }
32
33        /* vyvolání eventy */
34        for (ImageRecieveListener irl : listeners) {
35          irl.ImageRecieve(image);
36        }
37      }
38
39      else {
40        Log.e("Image", "Wrong Size");
41      }
42
43    }
```

Pro zobrazení obrazu se využívá funkcionality třídy BitmapFactory a její metody *decode()* . Ta přijímá jako vstup bajtové pole, ze kterého je následně vytvořena bitmapa a tu už je možné zobrazit v grafickém prvku ImageView. Tento proces je samozřejmě náročný na výpočetní výkon. Avšak při velikosti obrazu 640 x 480 px je telefon ještě schopen výpočty zpracovat.

5.6.7 Kompatibilita se staršími verzemi Androidu

Ač byla aplikace vyvíjena primárně pro Android ve verzi 4.0.3, je kompatibilní i se staršími verzemi Androidu. Emulátor systému Android umožňuje aplikaci otestovat jak na více verzích systému Android, tak i na větším počtu rozlišení displejů. Výchozím (a doporučeným) rozlišením pro tuto aplikace je 480 x 800 px. Pokud je aplikace spuštěna na displeji s nižším rozlišením, mezery mezi ovládacími prvky se zmenší a aplikace zůstává použitelná. Pokud se aplikace spustí na displejí s vyšším rozlišením, mezery mezi prvky se naopak zvětší.

5.7 Program pro ovládání robota

Klientský program, který ovládá robota, má pouze několik úkolů. Tím nejdůležitějším je umožnit připojení k serveru. Druhým úkolem je pak inicializace kamery a získávání obrazu. Vzhledem k nenáročnosti na nastavení byla aplikace od samého začátku koncipována jako konzolová - bez grafického uživatelského rozhraní. Po prvotním nastavení aplikace pracuje zcela autonomně, grafické rozhraní tak není potřeba. K ovládání robota je využito funkcionality knihovny IRobotCore, která byla námi (mnou a kolegou Martinem Sobotkou) vytvořena v rámci bakalářského projektu a až na pár drobných úprav nebyla měněna. Pro získání obrazu z kamery je používána volně dostupná knihovna AForge.NET.

5.7.1 Struktura aplikace

Následující seznam obsahuje všechny třídy, ze kterých se program skládá, spolu s jejich stručným popisem

- Program - Třída definující chování konzolového okna programu
- Paket - Definuje paket pro komunikaci se serverem
- Camera - Třída pro obsluhu webkamery
- Client - Třída pro komunikaci robota se serverem
- CameraClient - Třída pro komunikaci webkamery se serverem

- Orders - Třída definující příkazy pro robota
- Exceptions - Třída definující vlastní výjimky odchytávané v programu

5.7.2 Sockety v jazyce C#

Práce se sockety v jazyce C# je velmi obdobná jako v jazyce Java. Zde ale není použit serverový socket, ale klientský. Možnosti nastavení socketů v jazyce C# jsou mnohem bohatší než v jazyce Java. Konstruktor vypadá následovně:

- *public Socket(AddressFamily addressFamily, SocketType socketType,ProtocolType protocolType)*
  Prvním parametrem tohoto kostruktoru je proměnná typu AddressFamily. Typ AddressFamily umožňuje specifikovat, o jaký typ adresy se bude jednat (IPv4, IPv6, atd.). Druhým parametrem je SocketType. Socket může mít několik typů. V projektu je využíván typ Stream, který umožňuje dvoucestnou komunikaci s právě jedním uzlem přes. Posledním parametrem je ProtocolType. Ten specifikuje, jaký protokol se bude využívat pro samotnou komunikaci (např. TCP, UDP).
- *public Socket(SocketInformation socketInformation)*
  Tento kostruktor je prakticky totožný s tím, který je uveden výše, údaje ale nejsou předávány jednotlivě, ale uložené v jednom jediném datovém typu. Datový typ SocketInformation má dvě vlastnosti. Options a ProtocolInformation. Druhá jmenovaná proměnná sdružuje samotné informace o socketu. První pak umožňuje nastavit některé jeho vlastnosti.

5.7.3 Knihovna IRobotCore

Knihovna pro ovládání robota slouží jako prostředník mezi klientskou aplikací a robotem. Byla vytvořena v jazyce C# a nahrazuje původní rozhraní Open Interface. Hlavní nevýhodou Open Interface je, že se jedná o čistě číselné příkazy. Ty byly nahrazeny funkcemi a metodami zapouzdřenými ve třídě. Z hlediska funkcionality umožňuje třída IRobotCore všechno to, co umožňuje Open Interface. Z této třídy je v programu využívána pouze funkce pro pohyb

robota. Konkrétně funkce *Direction(short left, short right)* , kde left a right jsou rychlosti pro pravé a levé kolo robota. Při vytvoření instance třídy IRobotCore je robot připojen automaticky. Třída se stará i o udržování a korektní ukončení komunikace s robotem.

5.7.4 Grafická knihovna AForge.NET

Knihovna AForge.NET je zdarma dostupná knihovna pro práci s grafikou. Sestává z několika navzájem oddělených knihoven:

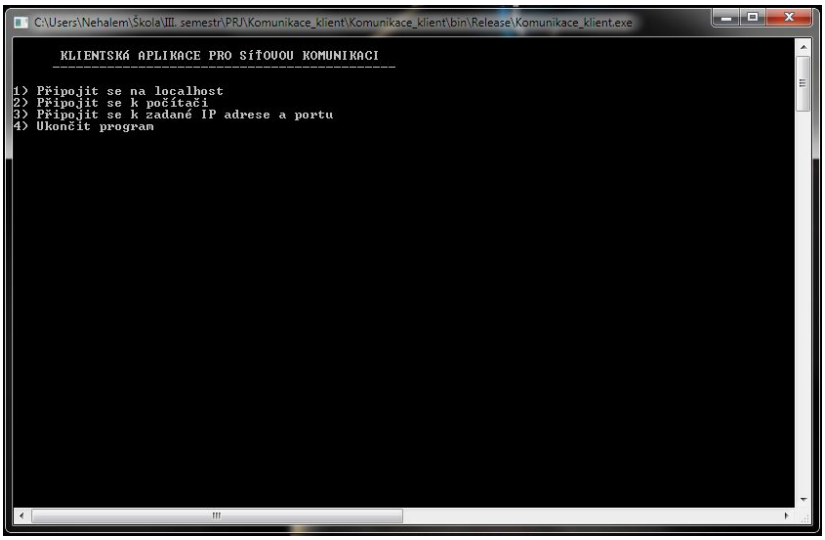
- AForge.Imaging - knihovna pro funkce zpracovávající obraz
- AForge.Vision - knihovna poskytující prostředky k počítačovému "vidění"
- AForge.Video - knihovna pro práci s videem
- AForge.Robotics - knihovna podporující některé robotické kity
- AForge.Neuro
- AForge.Genetic
- AForge.Fuzzy
- AForge.MachineLearning
- atd.

V programu je využívána knihovna AForge.NET jen k jedinému účelu, a to získání obrazu z webkamery. Tato knihovna dále umožňuje nastavit rozlišení obrazu a framerate (počet snímků za sekundu).

5.7.5 Uživatelské rozhraní aplikace

Jak již bylo výše uvedeno, aplikace je pouze konzolová a stará se o základní úkony: spojení s robotem a získání obrazu. Po zapnutí je uživateli nabídnuta možnost připojení na localhost, připojení k předem definovanému počítači a připojení k počítači se zadanou IP adresou a portem. Pokud se při připojení k serveru nebo k robotu vyskytnou chyby, uživatel je o

nich informován chybovou hláškou. Po úspěšném připojení k serveru má uživatel možnost zobrazení všech přijatých zpráv, odeslání zprávy, připojení kamery a odpojení se od serveru. Rozhraní aplikace je zobrazeno na obrázku 12.



Obrázek 12: Klientská aplikace

5.8 Součinnost serverové a klientské aplikace

Při komunikaci klientské a serverové aplikace je prvním limitujícím faktorem kvalita bezdrátové sítě. Zatímco v místech s dobrým pokrytím je rychlost komunikace dostačující, v místech se slabým signálem je situace o mnoho horší, což se projevuje mírným trháním obrazu. Důležitým faktem je také to, zda se jedná o bezdrátovou síť standardu g či n. Jak mobilní telefon, tak netbook mají možnost se k bezdrátové síti standardu n připojit.

Druhým limitujícím faktorem je výkon počítače ovládajícího robota. Při použití levných netbooků Asus Eee stouplo mnohdy využití procesoru nad hranici 80% a počítač se začal značně zahřívat. Při použití jakékoliv platformy výkonnější než první verze Intelu Atom nebyly žádné další problémy zaznamenány.

6 Závěr

V bakalářské práci se podařilo úspěšně zrealizovat všechny body zadání. Jako první byl, ve spolupráci s kolegy Brantem a Sobotkou, navržen komunikační protokol XML. Po jeho nasazení byl vytvořen druhý komunikační protokol, založený na posloupnosti čísel, a tento byl s protokolem XML porovnán. Vzhledem ke složitosti implementace a zpomalení výkonu celé aplikace při použití XML protokolu byl nakonec nasazen číselný protokol. Následně byla vytvořena aplikace pro ovládání robotické platformy iRobot Create pomocí operačního systému Android. Tato aplikace uživateli umožní ovládat robota jak tlačítky na displeji, tak mnohem intuitivnějším způsobem naklápěním telefonu. Současně byla přidána i obrazová data, aby byl uživatel informován o poloze robota a mohl jeho pohyb sledovat na displeji svého mobilního telefonu. Náročnost přenosu nekomprimovaných obrazových dat nakonec vedla k nutnosti použití komprese do formátu JPEG. Velikost obrazu tak klesla na čtvrtinu, což umožnilo přenést obraz s vyšším rozlišením. Aplikaci se podařilo upravit tak, aby jí bylo možné nainstalovat na většinu mobilních telefonů s operačním systémem Adnroid s různým rozlišením displeje.

Jako možnost rozšíření práce by mohla být použita optimalizace přenosu obrazových dat, efektivnější komprese a eliminace uživatelského rozhraní na počítači, který ovládá robota. Aplikace by také mohla být obohacena o hlasové povely, které však doposud nejsou programátorům aplikací v Androidu umožněny ze strany vývojářů tohoto systému.

Literatura

[1] AForge.NET: AForge.NET Framework [online]. `http://www.aforgenet.com/framework/docs/`, 2012, [cit. 2012-05-01].

[2] Android developers: The developer's Guide [online]. `http://doc.openrobotino.org/documentation/OpenRobotinoAPI/1/doc/rec_robotino_com/`, 2012, [cit. 2012-05-01].

[3] Garage, W.: OpenCSV v2.3 documentation [online]. `http://opensvt.itseez.com/`, 2012, [cit. 2012-05-01].

[4] Herout, P.: *Java - Bohatství knihoven*. České Budějovice: KOPP, 2006, ISBN 80-7232-288-5.

[5] Herout, P.: *Učebnice jazyka Java*. České Budějovice: KOPP, 2008, ISBN 978-80-7232-355-5.

[6] IRobot Corporation: IRobot Create Open Interface [online]. `http://chess.eecs.berkeley.edu/eecs124/iRobotDocs/CreateOpenInterface_v2.pdf`, 2006, [cit. 2012-05-01].

[7] IRobot Corporation: IRobot Create Owner's Guide [online]. `http://www.irobot.com/filelibrary/create/Create/20Manual_Final.pdf`, 2006, [cit. 2012-05-01].

[8] Micfosoft: MSDN Academic Alliance Program [online]. `http://www.microsoft.com/cze/education/licence/msdn_academic_alliance/default.mspx`, 2012, [cit. 2012-05-01].

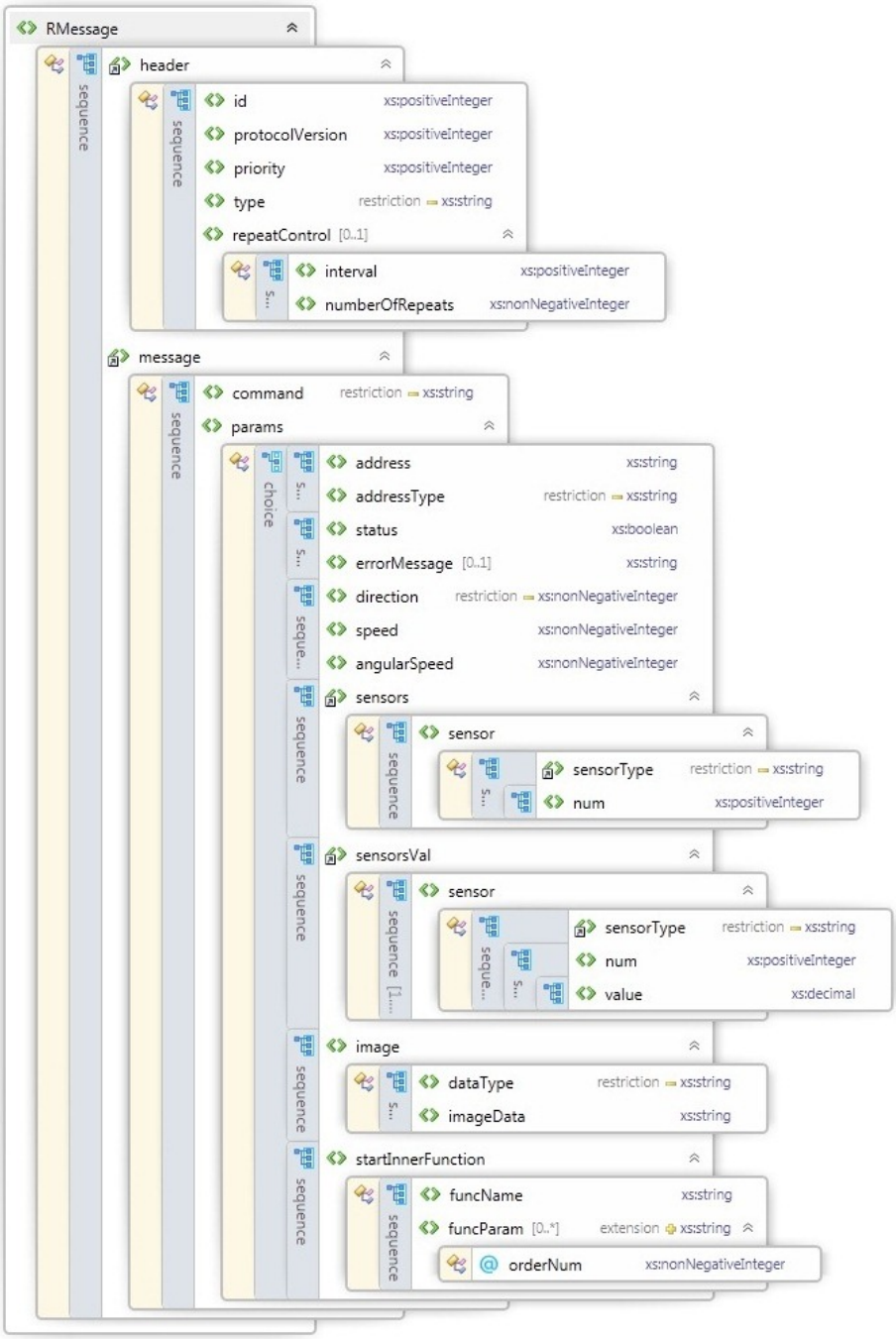
[9] Murphy, M. L.: *Android 2 průvodce programováním mobilních aplikací*. Brno: Computer Press, 2011, ISBN 978-80-251-3194-7.

[10] Nagel, C.; Evjen, B.: *C# 2008 - Programujeme profesionálně*. Brno: Computer Press, 2009, ISBN 978-80-251-2401-7.

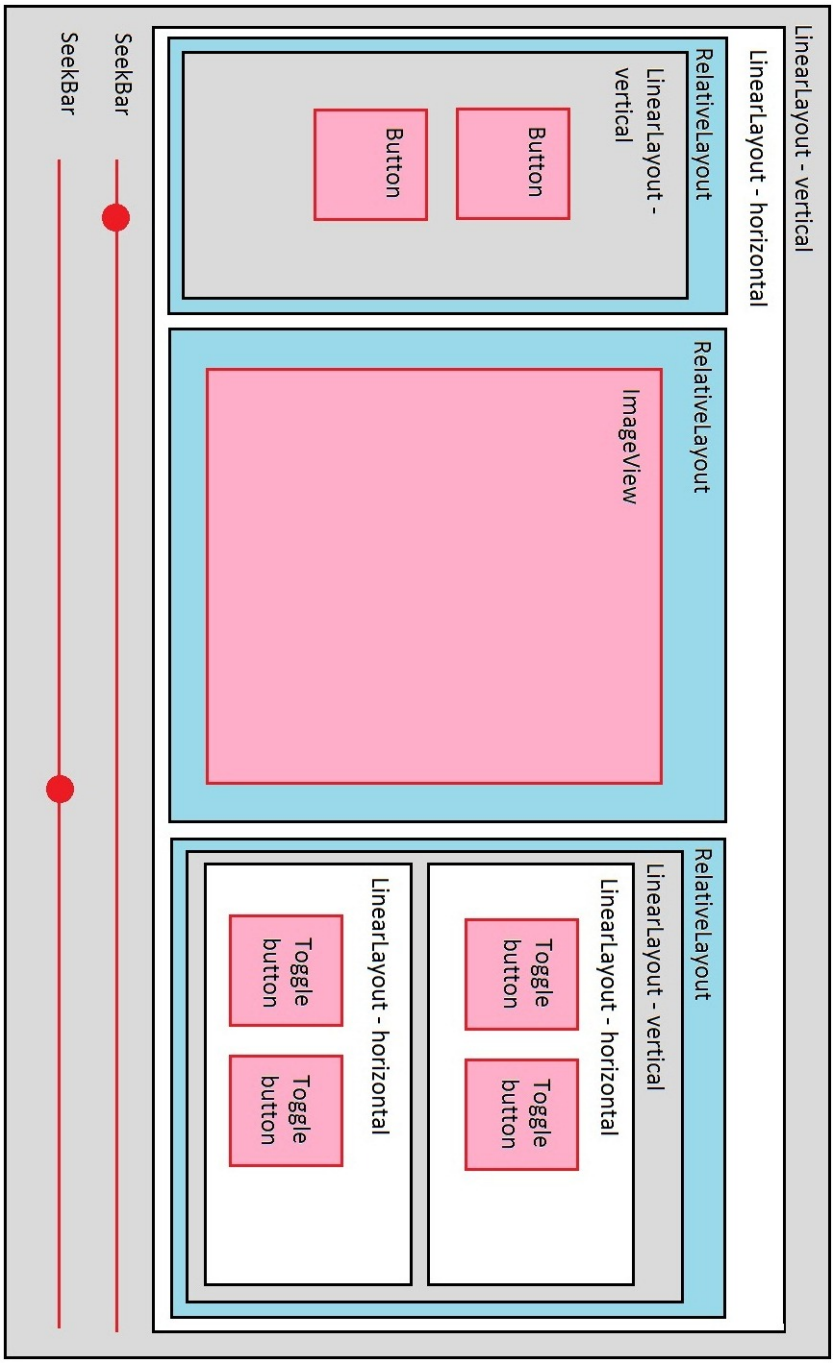
[11] Oracle: Java API Specification [online]. http://docs.oracle.com, 2012, [cit 2012-05-01].

[12] Sharp, J.: *Microsoft Visual C# Krok za krokem.* Brno: Computer Press, 2010, ISBN 978-80-251-31473.

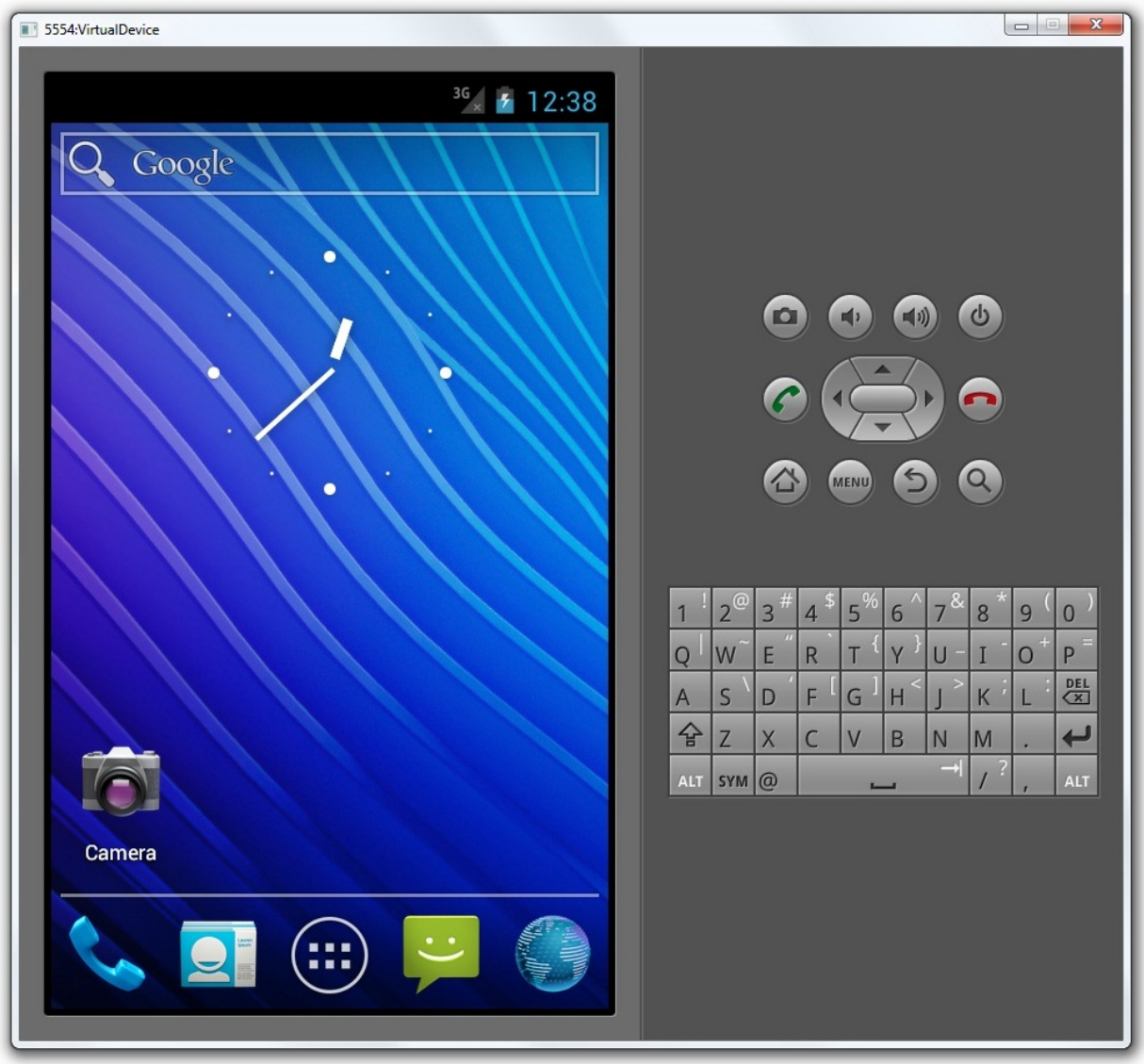
A Kompletní přehled položek XML protokolu



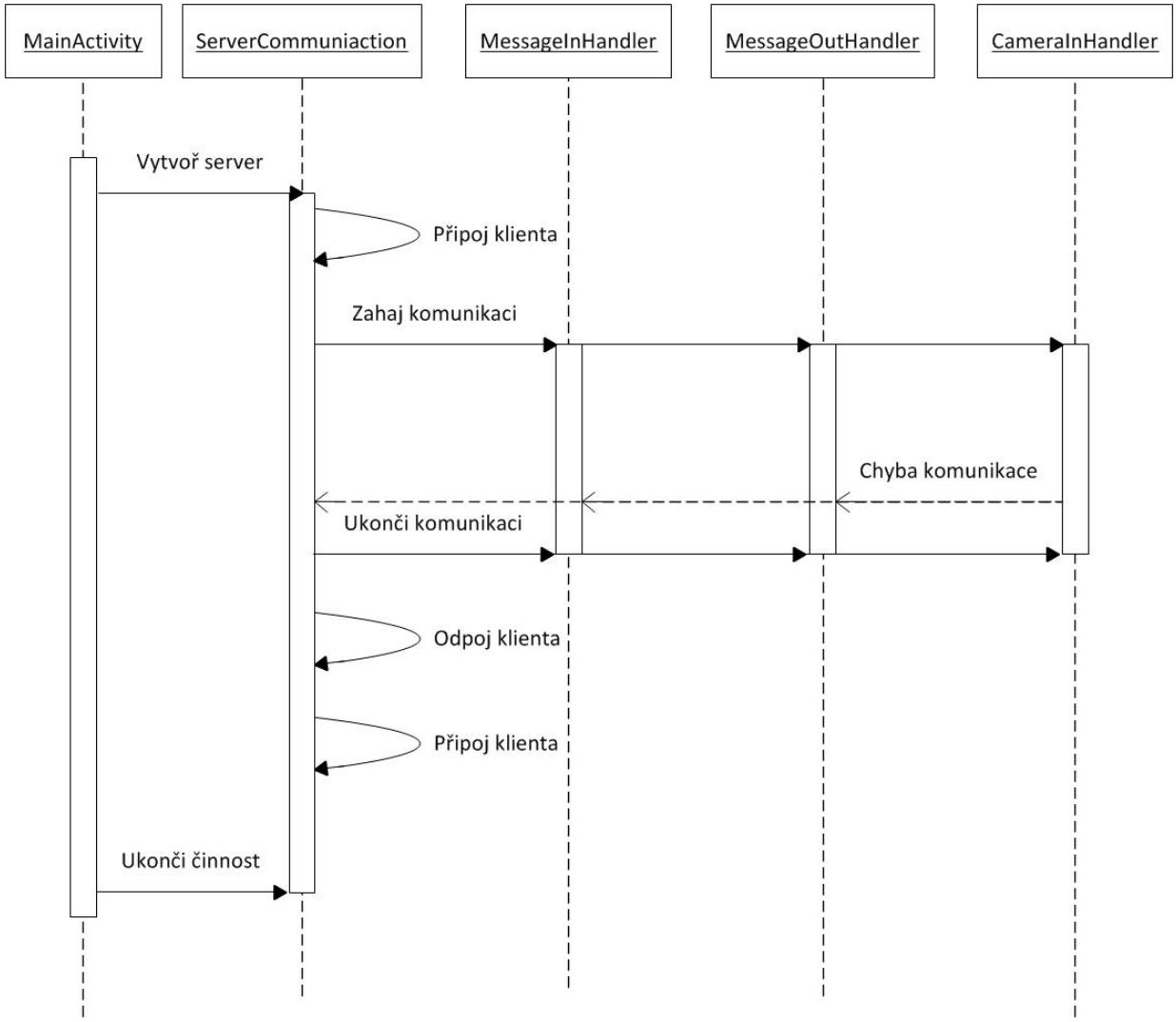
B Rozložení obrazovky hlavního uživatelského rozhraní



C Okno emulátoru systému Android



D Sekvenční diagram vláken aplikace



Poděkování: Tento materiál vznikl v rámci projektu ESF (CZ.1.07/2.2.00/07.0247)
Reflexe požadavků průmyslu na výuku v oblasti automatického řízení a měření.
Formát zpracování originálu: titulní list barevně, další listy včetně příloh barevně.